Mobile Systeme
Grundlagen und Anwendungen standortbezogener Dienste

Location Based Services in the Context of Web 2.0

Department of Informatics - MIN Faculty - University of Hamburg
Lecture Summer Term 2007

Dr. Thilo Horstmann

CLDC

NMEA

MIDP

Google Earth

OpenGIS

SQL

KML

Bluetooth

Mash-Ups

Web 2.0

J2ME

Loxodrome

Euler Spaces

RDMS

GPS

PostGIS

GPX

Maps

JSR 179

Polar Coordinates

API

Threads

Today: J2ME (IV)

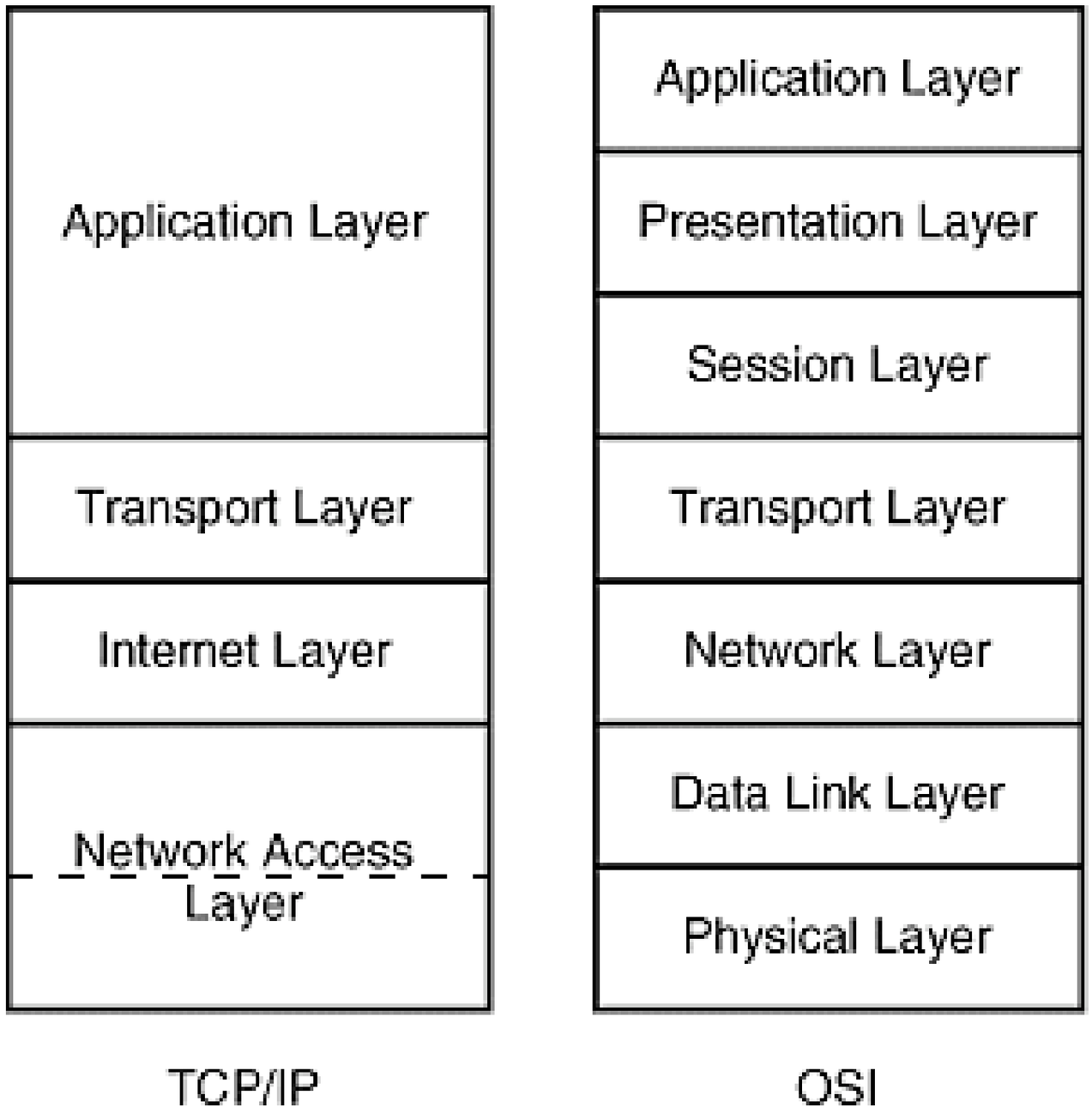
- Some network basics
- The J2ME Connection Framework
- Basic socket operations in J2ME

The 5 layer TCP/IP model

Layer	Protocols
Application	DHCP • DNS• FTP • Gopher • HTTP • IMAP4 • IRC • NNTP • XMPP • MIME • POP3 • SIP • SMTP • SNMP • SSH • TELNET • RPC • RTP • RTCP • TLS/SSL • SDP • SOAP • …
Transport	TCP • UDP • DCCP • SCTP • RSVP • GTP • …
Internet	P (IPv4 • IPv5 • IPv6) • IGMP • ICMP • BGP • RIP • OSPF • ISIS • IPsec • ARP • RARP • …
Data Link	802.11 • ATM • DTM • Ethernet • FDDI • Frame Relay • GPRS • EVDO • HSPA • HDLC • PPP • L2TP • PPTP • …
Physical	Ethernet physical layer • ISDN • Modems • PLC • SONET/SDH • G.709 • …

TCP/IP and OSI

- OSI is a de-jure standard and TCP/IP is a de-facto standard.
- The TCP/IP world agrees on a protocol standard which can be made to work in diverse heterogeneous networks.
- The OSI world is more interested in the standard than the implementation.



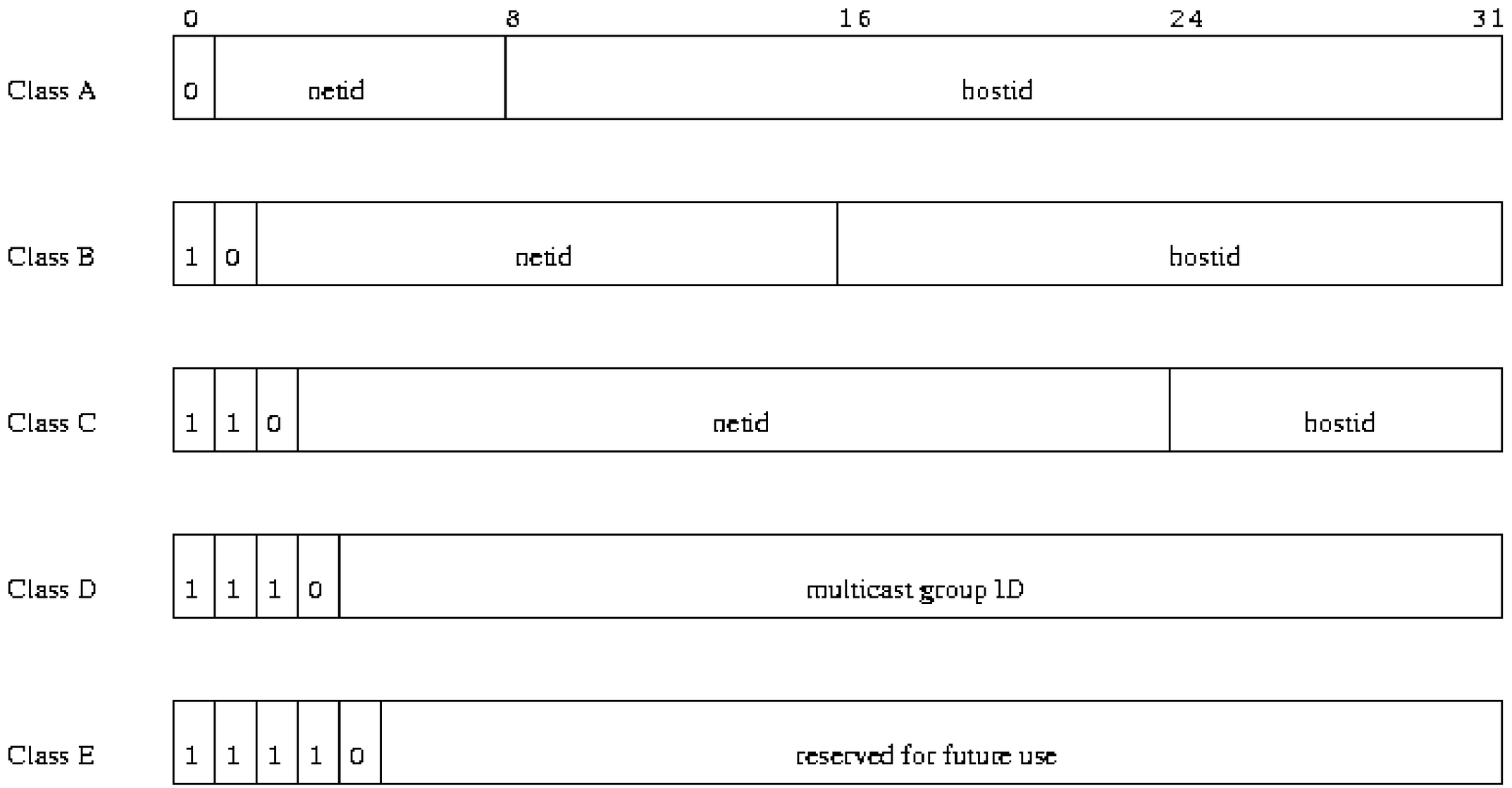
Internet Protocol (IP)

- IP is the network layer
  - packet delivery service (host-to-host).
  - translation between different data-link protocols.
  - it must be capable of providing communication between hosts on different kinds of networks (different data-link implementations).
- IP provides connectionless, unreliable delivery of IP datagrams.
  - Connectionless: each datagram is independent of all others.
  - Unreliable: there is no guarantee that datagrams are delivered correctly or even delivered at all.

IP addresses

- IP addresses are logical addresses (not physical)
- 32 bits (IPv4).
- Includes a network ID and a host ID.
- Every host must have a unique IP address.
- A single network interface is assigned a single IP address called the host address.
- A host may have multiple interfaces, and therefore multiple host addresses.
- Hosts that share a network all have the same IP network address (the network ID).

IP netid & hostid



IP Network classes

- Class A addresses (range from 0.0.0.0 to 127.255.255.255) are for extremely large networks containing many millions of hosts.
- Class B addresses (range from 128.0.0.0 to 191.255.255.255) are for medium to large networks containing many thousands of hosts.
- Class C addresses (range from 192.0.0.0 to 223.255.255.255) are for small networks with a couple of hundred hosts.
- Class D addresses (range from 224.0.0.0 to 239.255.255.255) are multicast identifying a group of hosts.

Transmission Control Protocol (TCP)

- Connection-oriented
  - a virtual connection is established before any user data is transferred.
  - If the connection cannot be established - the user program is notified (finds out).
  - If the connection is ever interrupted - the user program(s) is finds out there is a problem.
- Reliable
  - every transmission of data is acknowledged by the receiver
  - If the sender does not receive acknowledgment within a specified amount of time, the sender retransmits the data.

Transmission Control Protocol (cont.)

- Byte-Stream
  - connection is treated as a stream of bytes.
  - The user application does not need to package data in individual datagrams (as with UDP)
- Full-duplex
  - TCP provides transfer in both directions (over a single virtual connection).
  - To the application program these appear as 2 unrelated data streams, although TCP can piggyback control and data communication by providing control information (such as an ACK) along with user data.

User Datagram Protocol (UDP)

- UDP is a minimal message-oriented transport layer protocol (RFC 768)
- provides a very simple interface between a network layer and an application layer
- UDP uses ports to allow application-to-application communication
- UDP supports packet broadcast (sending to all on local network) and multicasting (send to all subscribers).
- UDP provides no guarantees to the upper layer protocol for message delivery

Adressing in TCP/IP

- TCP/IP denotes a suite of protocols including TCP, IP and UDP
- Each TCP/IP address includes:
    - Internet Address
    - Protocol (UDP or TCP)
    - Port Number
- a port is a special number present in the header of a data packet.
    - a server used for sending and receiving email may provide both an SMTP and a POP3 service; these will be handled by different server processes, and the port number will be used to determine which data is associated with which process.

Summary: TCP/IP

- IP: network layer protocol
    - unreliable datagram delivery between hosts.
- UDP: transport layer protocol
    - unreliable datagram delivery between processes.
- TCP: transport layer protocol
    - reliable, byte-stream delivery between processes.

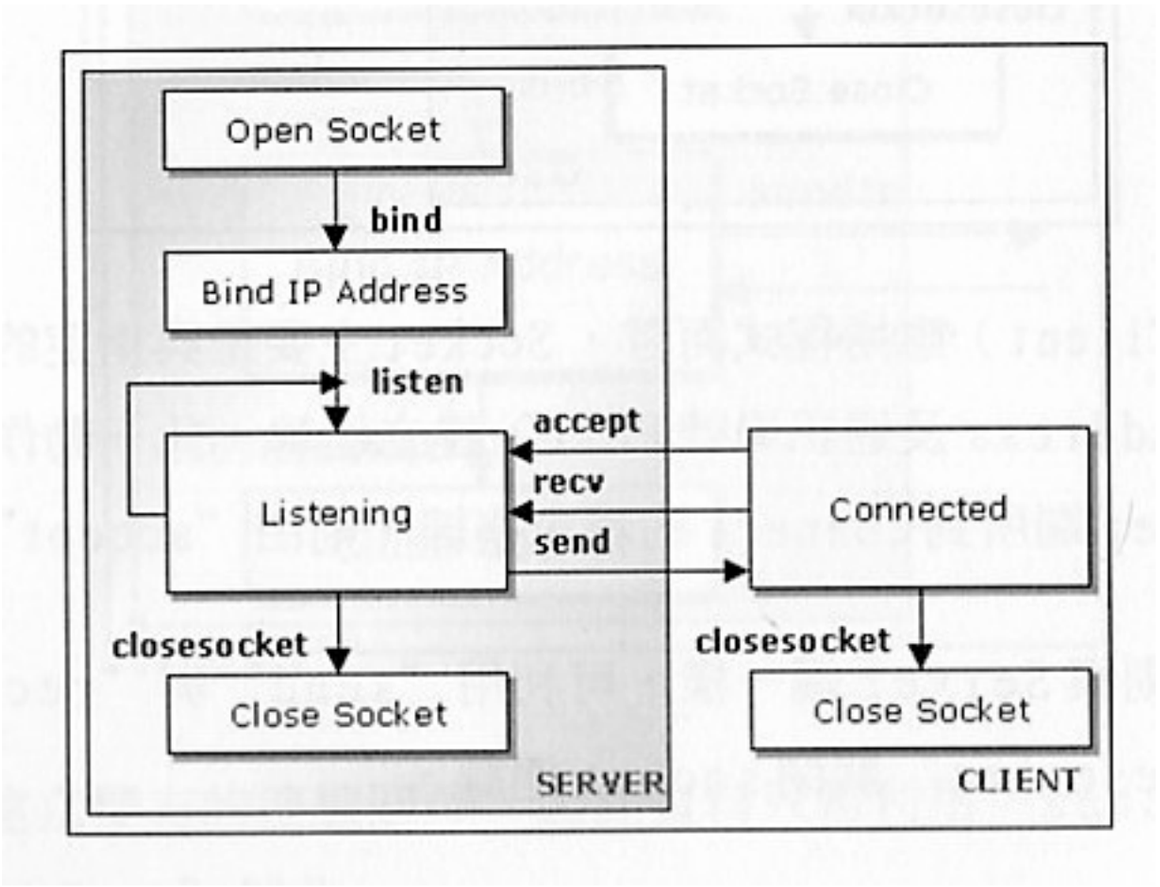
Sockets

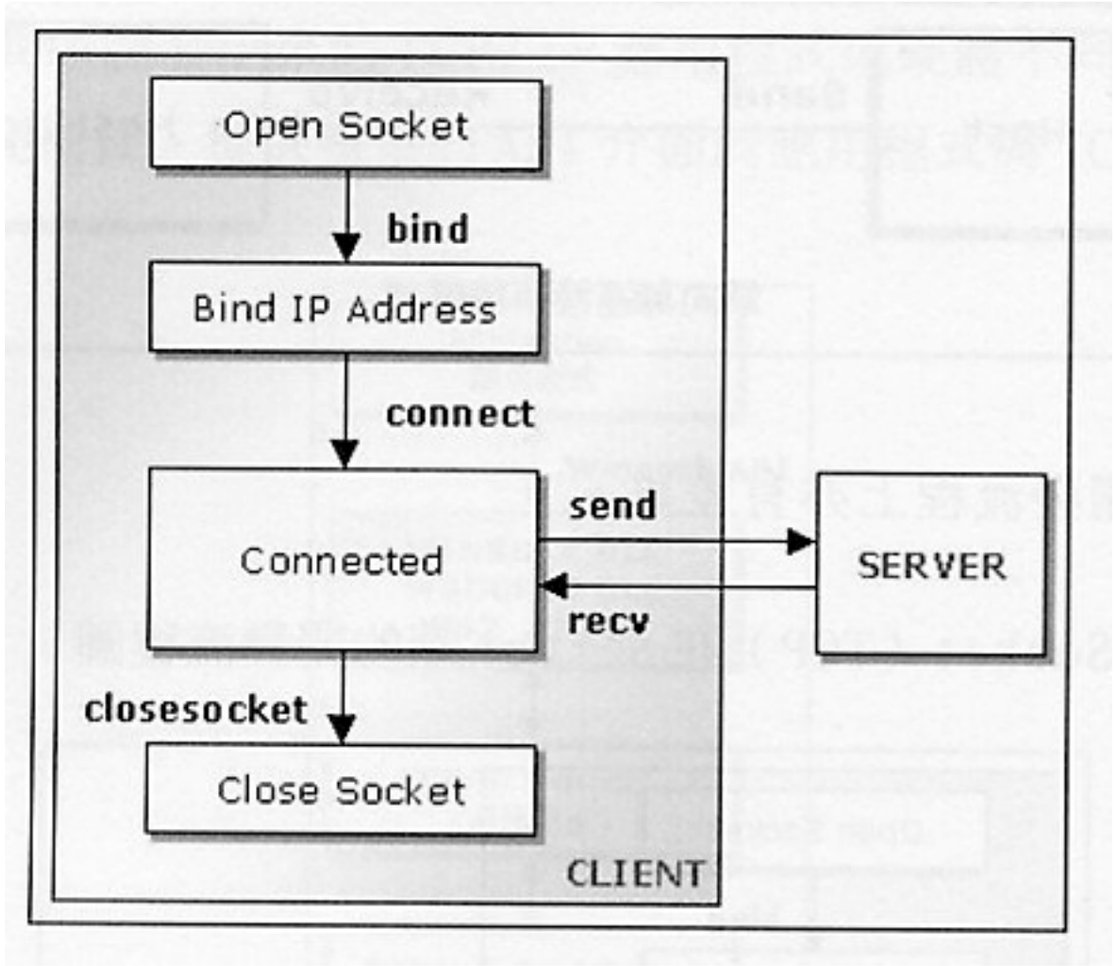
- TCP/IP does not include an API definition.
- There are a variety of APIs for use with TCP/IP: Sockets, Winsock, MacTCP
- Functions needed:
    - Specify local and remote communication endpoints
    - Initiate a connection
    - Wait for incoming connection
    - Send and receive data
    - Terminate a connection gracefully
    - Error handling

(Berkeley) Sockets

- A socket is an abstract representation of a communication endpoint.
- Sockets work with Unix I/O services just like files, pipes & FIFOs.
- Sockets (obviously) have special needs:
    - establishing a connection
    - specifying communication endpoint addresses
- Formally a socket is defined by a group of four numbers, these are
- The remote host identification number or address
- The remote host port number
- The local host identification number or address
- The local host port number

Creating a TCP socket connection (server (l.), client (r.)



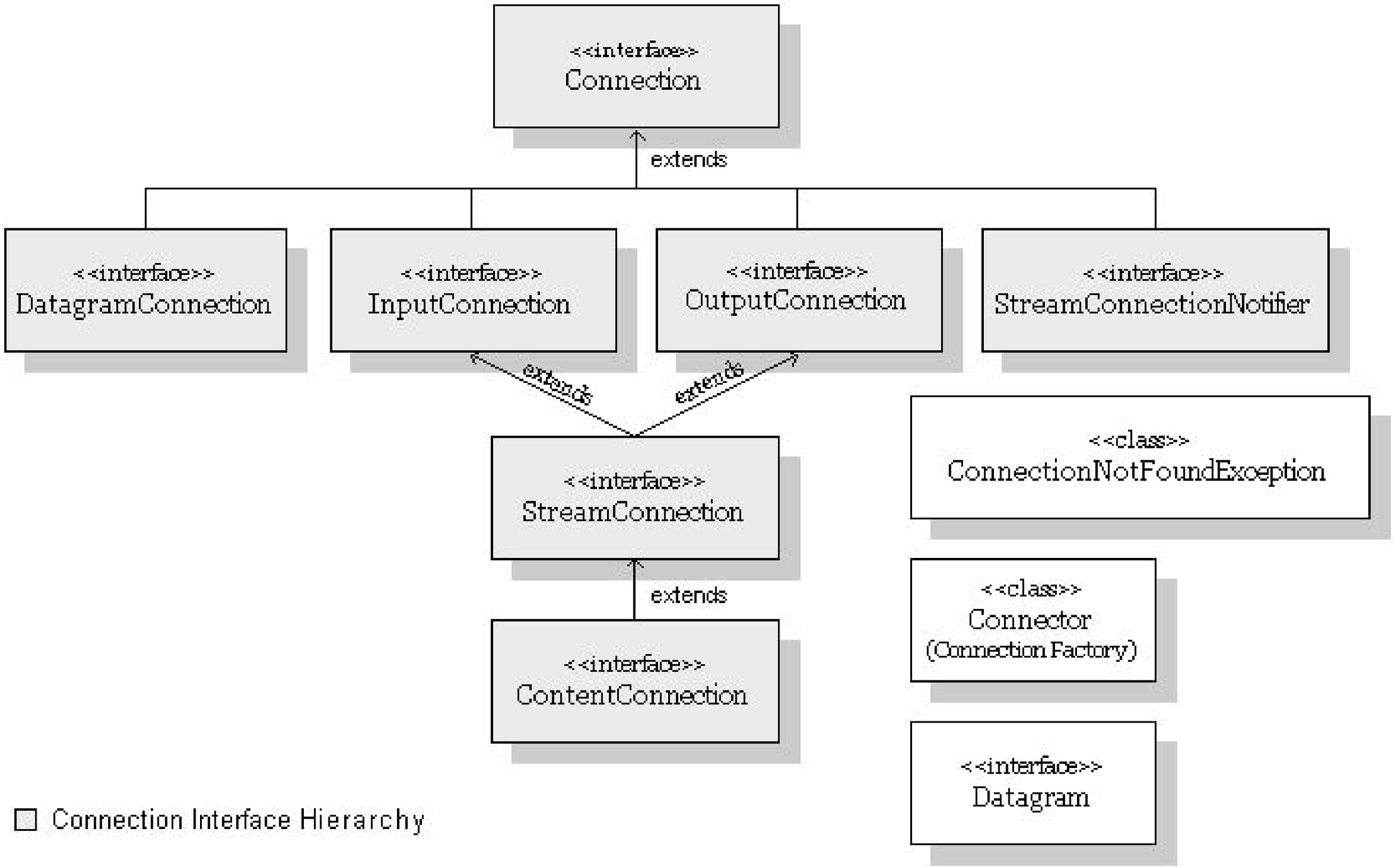


The CLDC Generic Connection Framework (GCF)

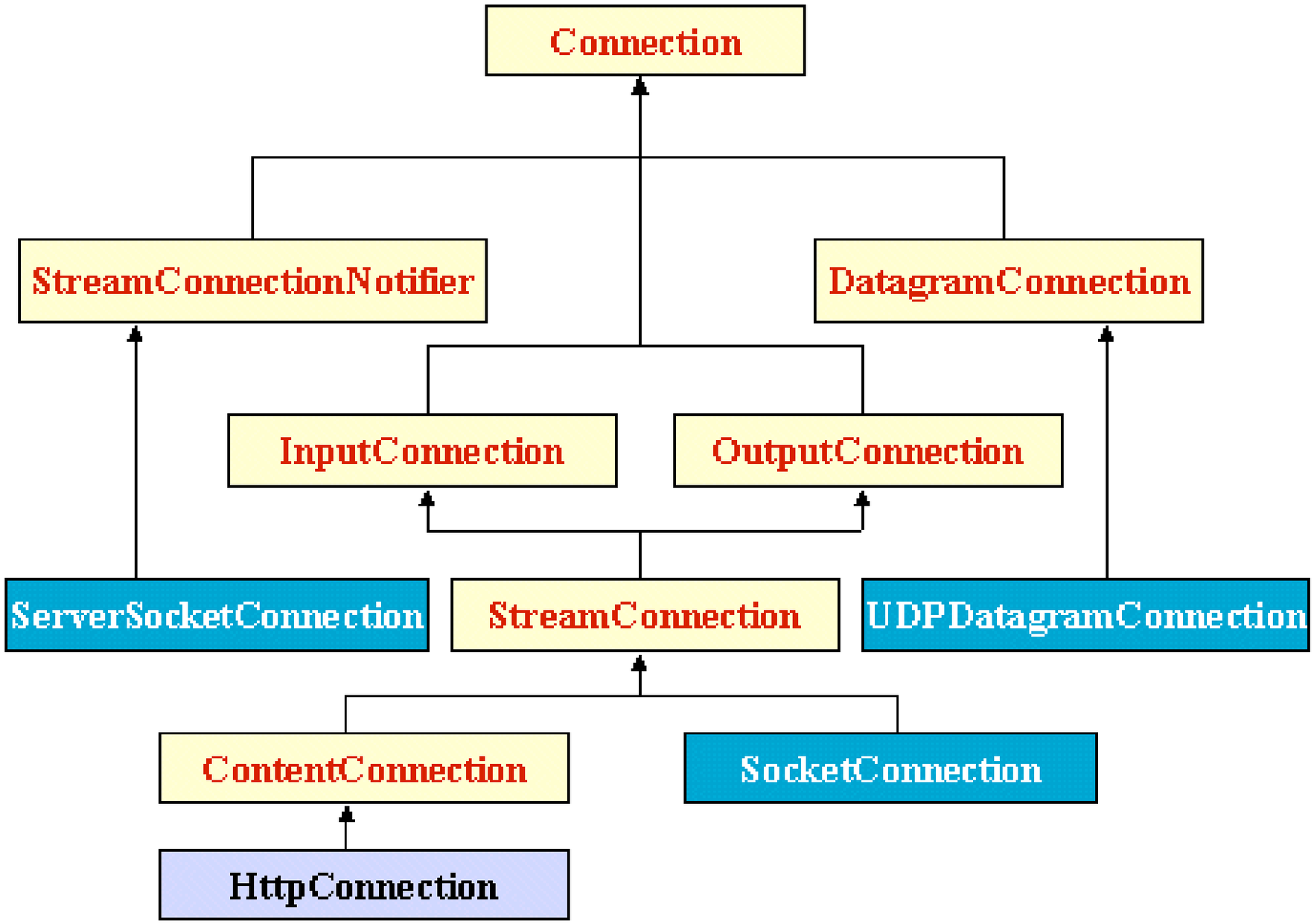
Overview: The GCF (in javax.microedition.io)

- J2SE java.net and java.io APIs were considered too large to fit into the constrained memory available in mobile devices.
- The GCF is a straightforward hierarchy of interfaces and classes to create connections (such as HTTP, datagram, or streams) and perform I/O.
- GCF provides a common *foundation* API for all the basic connection types - for packet-based (data blocks) and stream-based (contiguous or sequence of data) input and output.
- This generalization is possible through the use of:
  - An interface hierarchy that is extensible
  - A connection factory
  - Standard Uniform Resource Locators (URLs) to indicate the connection types to create

The CLDC GCF Interface Hierarchy



GCF Interfaces (CLDC) & Implementation (MIDP)



The Connector class

- The creation of connection will be archived using the Connector class (javax.microedition.io.Connector)
- This class is factory for creating new Connection objects
- creation of Connections is performed dynamically by looking up a protocol implementation class
  - whose name is formed from the platform name (read from a system property)
  - and the protocol name of the requested connection (extracted from the parameter string supplied by the application programmer.)

URIs identify connection types and endpoints

- scheme://user:password@host:port/url-path;parameters, where:
  - scheme specifies the access method or protocol, such as FTP or HTTPS. In the GCF, it describes the connection type to use, which maps to an underlying connection or I/O protocol.
  - user is an optional user name.
  - password is an optional password.
  - host is the fully qualified name or the IP address of the host where the resource is located.
  - port is an optional port to use. Its interpretation depends on the scheme.
  - url-path is the "path" to the resource. Its format and interpretation depend on the scheme. The url-path may define optional parameters.

Required and optional URI Schemes of the GCF

URL Scheme	Connectivity	GCF Connection Type	Defined By...
btl2cap	Bluetooth	L2CAPConnection	JSR 82. Support is optional.
datagram	Datagram	DatagramConnection	All CLDC- and CDC-based profiles, such as MIDP, Foundation and related profiles, and with JSR 197, J2SE. Support is optional.
file	File Access	FileConnection, InputConnection	JSR 75. Support is optional.
http	HyperText Transport Protocol	HttpConnection	MIDP 1.0, MIDP 2.0, Foundation Profile, J2SE (JSR 197). Support is required.
https	Secure HTTP	HttpsConnection	MIDP 2.0. Support is required.
comm	Serial I/O	CommConnection	MIDP 2.0. Support is optional.
sms, mms, cbs	Short Messaging Service, Multimedia Messaging Service, Cell Broadcast SMS	MessageConnection	JSR 120, JSR 205. Support is optional.
apdu, jcrmi	Security Element	APDUConnection, JavaCardRMIConnection	JSR 177. Support is optional.
socket, serversocket	Socket	SocketConnection, ServerSocketConnection	JSR118 (MIDP 2.0). Support is optional.
datagram	UDP Datagram	UDPDatagramConnection	JSR118 (MIDP 2.0). Support is optional.

Core GCF Connection Types, By Profile

GCF Interfaces, Classes, Exceptions	Vendors Must Support?	CLDC 1.0, 1.1	MIDP 1.0	MIDP 2.0	Foundation and Related Profiles	GCF for J2SE (JSR 197)
CommConnection	N			X		
Connection (base connection)	Y	X	X	X	X	X
Content Connection	Y	X	X	X	X	X
Datagram Connection	N	X	X	X	X	X
HttpConnection	Y		X	X	X	X
HttpsConnection	Y			X		
InputConnection	Y	X	X	X	X*	X*
OutputConnection	Y	X	X	X	X	X
SecureConnection	N			X		
ServerSocket Connection	N			X		
SocketConnection	N			X		
StreamConnection	Y	X	X	X	X	X
StreamConnection Notifier	Y	X	X	X	X	X
UDPDatagram Connection	N			X		

Create a connection using the Connector class

```
...
String url = "socket://www.j2medeveloper.com:80";
...
SocketConnection c = null;
InputStream s = null;
try {
    c = (SocketConnection)Connector.open(url);
    s = c.openInputStream();
    ...
    // Read from the input stream, handle input data.
    ...
}
```

Create a connection using the Connector class

```
...catch (ConnectionNotFoundException cne) {
    // Connection specified in URL can't be created.
    // Handle exception, throw exception or return error.
} catch (IllegalArgumentException iae) {
    // One of the arguments is in error.  In this example, the
    // only argument to open is URL, so the only expected
    // exceptions are ConnectionNotFoundException or IOException.
    // Handle exception, throw exception or return error.
} catch (IOException ioe) {
    // Handle exception, throw exception or return error.
} finally {
    try {
        if (s != null) s.close();
        if (c != null) c.close();
    } catch (Exception e) {
        // Handle exception, throw exception or return error.
    }
}
...
```

Examples of the creation of various connection types:

```
String url = "socket://www.j2medeveloper.com:80";
SocketConnection c = (SocketConnection)Connector.open(url);

String url = "http://www.j2medeveloper:80/com/myServlet";
HttpConnection c = (HttpConnection)Connector.open(url);

String url = "file:///myResourceFile.res";
InputConnection c = (InputConnection)Connector.open(url);

String url = "file:///myResourceFile.res";
FileConnection c = (FileConnection)Connector.open(url);

String url = "datagram://www.j2medeveloper.com:7001";
UDPDatagramConnection c = (UDPDatagramConnection)
Connector.open(url);
```

Socket communication: The server

```
... // create a server to listen on port 2500
ServerSocketConnection server = (ServerSocketConnection) Connector.open
("socket://:2500");
// wait for a connection
SocketConnection client = (SocketConnection) server.acceptAndOpen();
// set application-specific options on the socket;
// call setSocketOption to set other options
client.setSocketOption(DELAY, 0);
client.setSocketOption(KEEPALIVE, 0);
// open streams
DataInputStream dis = client.openDataInputStream();
DataOutputStream dos = client.openDataOutputStream();
// read client request
String result = is.readUTF();
// process request and send response
os.writeUTF(...);
// close streams and connections
is.close();
os.close();
client.close();
server.close();
...
```

Socket communication: The client

```
...
SocketConnection client = (SocketConnection) Connector.open("socket://" +
hostname + ":" + port);
// set application-specific options on the socket. Call setSocketOption to set
other options
client.setSocketOption(DELAY, 0);
client.setSocketOption(KEEPALIVE, 0);
InputStream is = client.openInputStream();
OutputStream os = client.openOutputStream();
// send something to server
os.write("some string".getBytes());
// read server response
int c = 0;
while((c = is.read()) != -1) {
   // do something with the response
}
// close streams and connection
is.close();
os.close();
client.close();
...
```

Socket communication: accessing a web server

```
// establish a socket connection with remote server
streamConnection = (StreamConnection) Connector.open(connectString);
// create DataOuputStream on top of the socket connection
outputStream = streamConnection.openOutputStream();
dataOutputStream = new DataOutputStream(outputStream);
// send the HTTP request. HTTP/1.1 requires to send the Host header!
dataOutputStream.writeChars("GET /index.htm HTTP/1.0 \n");
dataOutputStream.flush();
// create DataInputStream on top of the socket connection
inputStream = streamConnection.openInputStream();
dataInputStream = new DataInputStream(inputStream);
// retrieve the contents of the requested page from Web server
int inputChar;
while ( (inputChar = dataInputStream.read()) != -1) {
   results.append((char) inputChar);
}
// display the page contents on the phone screen
...
} catch (IOException e) {
      System.err.println("Exception caught:" + e);
} finally {
      // free up I/O streams and close the socket connection
```

This Lecture

- Network:
  - Andrew S. Tanenbaum, Computer Networks, 4th Edition, Prentice Hall ISBN-13: 978-0-13-066102-9
- Generic Connection Framework:
  - http://developers.sun.com/techtopics/mobility/midp/articles/genericframework/
  - http://developers.sun.com/techtopics/mobility/midp/articles/midp2network/

Thank you!

Dr. Thilo Horstmann

e-mail: thilo.horstmann@gmail.com
blog: http://www.das-zentralorgan.de